NAG Library Routine Document

E05JCF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

E05JCF may be used to supply optional parameters to E05JBF from an external file. The initialization routine E05JAF **must** have been called before calling E05JCF.

2 Specification

```
SUBROUTINE E05JCF (IOPTS, COMM, LCOMM, IFAIL)

INTEGER            IOPTS, LCOMM, IFAIL
REAL (KIND=nag_wp) COMM(LCOMM)
```

3 Description

E05JCF may be used to supply values for optional parameters to E05JBF. E05JCF reads an external file and each line of the file defines a single optional parameter. It is only necessary to supply values for those parameters whose values are to be different from their default values.

Each optional parameter is defined by a single character string, of up to 72 characters, consisting of one or more items. The items associated with a given optional parameter must be separated by spaces, or equals signs [=]. Alphabetic characters may be upper or lower case. The string

```
Static Limit = 100
```

is an example of a string used to set an optional parameter. For each optional parameter the string contains one or more of the following items:

- a mandatory keyword;

- a phrase that qualifies the keyword;

- a number that specifies an integer or real value. Such numbers may be up to 16 contiguous characters.

Blank strings and comments are ignored. A comment begins with an asterisk (*) and all subsequent characters in the string are regarded as part of the comment.

The implied data type (character, integer or real) of each value to set **must** match that expected by the corresponding optional parameter.

The file containing the optional parameters must start with `Begin` and must finish with `End`. An example of a valid options file is:

```
Begin * Example options file
   Static Limit = 500
End
```

Optional parameter settings are preserved following a call to E05JBF and so the keyword **Defaults** is provided to allow you to reset all the optional parameters to their default values before a subsequent call to E05JBF.

A complete list of optional parameters, their symbolic names and default values is given in Section 11 in E05JBF.

4 References

None.

5 Parameters

1: IOPTS – INTEGER *Input*

On entry: the unit number of the option file to be read.

Constraint: IOPTS is a valid unit open for reading.

2: COMM(LCOMM) – REAL (KIND=nag_wp) array *Communication Array*

On exit: COMM **must not** be altered between calls to any of the routines E05JBF, E05JCF, E05JDF, E05JEF, E05JFF, E05JGF, E05JHF, E05JJF, E05JKF and E05JLF.

3: LCOMM – INTEGER *Input*

On entry: the dimension of the array COMM as declared in the (sub)program from which E05JCF is called.

Constraint: LCOMM ≥ 100.

4: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, LCOMM < 100,
or the initialization routine E05JAF has not been called.

IFAIL $= 2$

Could not read options file on unit IOPTS. This may be because:

(a) IOPTS is not a valid unit number;

(b) a file is not associated with unit IOPTS, or if it is, is unavailable for read access;

(c) one or more lines of the options file are invalid. A keyword or keyword combination was not recognized;

(d) `Begin` was found, but end-of-file was found before `End` was found;

(e) end-of-file was found before `Begin` was found.

In cases (c) and (d) all optional parameters that were set from the file before the error was encountered will remain set on exit.

IFAIL $= 3$

An option value to be set was out of range with respect to the corresponding optional parameter given. See Section 11 in E05JBF for allowable values of the optional parameters.

IFAIL = 5

> A supplied string associated with a numeric value could not be parsed. Check that all such strings specify valid integer or real values.

7 Accuracy

Not applicable.

8 Further Comments

E05JDF, E05JEF, E05JFF or E05JGF may also be used to supply optional parameters to E05JBF.

9 Example

This example finds the global minimum of the 'peaks' function in two dimensions

$$F(x,y) = 3(1-x)^2 \exp\left(-x^2-(y+1)^2\right) - 10\left(\frac{x}{5}-x^3-y^5\right)\exp\left(-x^2-y^2\right) - \frac{1}{3}\exp\left(-(x+1)^2-y^2\right)$$

on the box $[-3,3] \times [-3,3]$.

The function F has several local minima and one global minimum in the given box. The global minimum is approximately located at $(0.23,-1.63)$, where the function value is approximately -6.55.

By specifying an initialization list via LIST, NUMPTS and INITPT we can start E05JBF looking close to one of the local minima and check that it really does move away from that point to one of the global minima.

More precisely, we choose $(-1,0)$ as our initial point (see Section 9.3), and let the initialization list be

$$\begin{pmatrix} -3 & -1 & 3 \\ -3 & 0 & 3 \end{pmatrix}.$$

This example solves the optimization problem using some of the optional parameters described in Section 11 in E05JBF.

9.1 Program Text

```
!   E05JCF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module e05jcfe_mod

!     E05JCF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
      Use nag_library, Only: x04baf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                   :: lcomm = 100, nin = 5,           &
                                              ninopt = 7, nout = 6
!     .. Local Scalars ..
      Logical                              :: plot
    Contains
      Subroutine outbox(n,boxl,boxu)

!       Displays edges of box with bounds BOXL and BOXU in format suitable
!       for plotting.

!       .. Scalar Arguments ..
        Integer, Intent (In)                 :: n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: boxl(n), boxu(n)
```

```
!       .. Local Scalars ..
        Character (80)                        :: rec
!       .. Executable Statements ..
        Write (rec,99999) boxl(1), boxl(2)
        Call x04baf(nout,rec)
        Write (rec,99999) boxl(1), boxu(2)
        Call x04baf(nout,rec)
        Write (rec,'()')
        Call x04baf(nout,rec)
        Write (rec,99999) boxl(1), boxl(2)
        Call x04baf(nout,rec)
        Write (rec,99999) boxu(1), boxl(2)
        Call x04baf(nout,rec)
        Write (rec,'()')
        Call x04baf(nout,rec)
        Write (rec,99999) boxl(1), boxu(2)
        Call x04baf(nout,rec)
        Write (rec,99999) boxu(1), boxu(2)
        Call x04baf(nout,rec)
        Write (rec,'()')
        Call x04baf(nout,rec)
        Write (rec,99999) boxu(1), boxl(2)
        Call x04baf(nout,rec)
        Write (rec,99999) boxu(1), boxu(2)
        Call x04baf(nout,rec)
        Write (rec,'()')
        Call x04baf(nout,rec)

        Return

99999   Format (F20.15,1X,F20.15)
      End Subroutine outbox
      Subroutine objfun(n,x,f,nstate,iuser,ruser,inform)

!       Routine to evaluate E05JBF objective function.

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Out)      :: f
        Integer, Intent (Out)                 :: inform
        Integer, Intent (In)                  :: n, nstate
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout)    :: ruser(*)
        Real (Kind=nag_wp), Intent (In)       :: x(n)
        Integer, Intent (Inout)               :: iuser(*)
!       .. Local Scalars ..
        Real (Kind=nag_wp)                    :: x1, x2
        Character (80)                        :: rec
!       .. Intrinsic Procedures ..
        Intrinsic                             :: exp
!       .. Executable Statements ..
        inform = 0

        If (inform>=0) Then

!         If INFORM >= 0 then we're prepared to evaluate OBJFUN
!         at the current X

          If (nstate==1) Then

!           This is the first call to OBJFUN

            Write (rec,'()')
            Call x04baf(nout,rec)
            Write (rec,99999)
            Call x04baf(nout,rec)
          End If

          x1 = x(1)
          x2 = x(2)

          f = 3.0E0_nag_wp*(1.0E0_nag_wp-x1)**2*exp(-(x1**2)-(x2+1.0E0_nag_wp) &
            **2) - 1.0E1_nag_wp*(x1/5.0E0_nag_wp-x1**3-x2**5)*exp(-x1**2-x2**2 &
            ) - 1.0E0_nag_wp/3.0E0_nag_wp*exp(-(x1+1.0E0_nag_wp)**2-x2**2)
        End If

        Return

99999   Format (1X,'(OBJFUN was just called for the first time)')
      End Subroutine objfun
      Subroutine monit(n,ncall,xbest,icount,ninit,list,numpts,initpt,nbaskt, &
        xbaskt,boxl,boxu,nstate,iuser,ruser,inform)

!       Monitoring routine for E05JBF.

!       .. Scalar Arguments ..
        Integer, Intent (Out)                :: inform
        Integer, Intent (In)                 :: n, nbaskt, ncall, ninit, nstate
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: boxl(n), boxu(n),              &
                                                list(n,ninit),                 &
                                                xbaskt(n,nbaskt), xbest(n)
        Real (Kind=nag_wp), Intent (Inout)   :: ruser(*)
        Integer, Intent (In)                 :: icount(6), initpt(n), numpts(n)
        Integer, Intent (Inout)              :: iuser(*)
!       .. Local Scalars ..
        Integer                              :: i, j
        Character (80)                       :: rec
!       .. Executable Statements ..
        inform = 0

        If (inform>=0) Then

!         We are going to allow the iterations to continue.

          If (nstate==0 .Or. nstate==1) Then

!           When NSTATE.EQ.1, MONIT is called for the first time. When
!           NSTATE.EQ.0, MONIT is called for the first AND last time.
!           Display a welcome message

            Write (rec,'()')
            Call x04baf(nout,rec)
            Write (rec,99999)
            Call x04baf(nout,rec)
            Write (rec,'()')
            Call x04baf(nout,rec)

            If (plot .And. (n==2)) Then
              Write (rec,99998)
              Call x04baf(nout,rec)
              Write (rec,'()')
              Call x04baf(nout,rec)
            End If

          End If

          If (plot .And. (n==2)) Then

!           Display the coordinates of the edges of the current search
!           box

            Call outbox(n,boxl,boxu)

          End If

          If (nstate<=0) Then

!           MONIT is called for the last time

            If (plot .And. (n==2)) Then
              Write (rec,99997)
              Call x04baf(nout,rec)
              Write (rec,'()')
              Call x04baf(nout,rec)
            End If

            Write (rec,99996) icount(1)
            Call x04baf(nout,rec)
            Write (rec,99995) ncall
            Call x04baf(nout,rec)
            Write (rec,99994) icount(2)
            Call x04baf(nout,rec)
            Write (rec,99993) icount(3)
            Call x04baf(nout,rec)
            Write (rec,99992) icount(4)
            Call x04baf(nout,rec)
            Write (rec,99991) icount(5)
            Call x04baf(nout,rec)
            Write (rec,99990) icount(6)
            Call x04baf(nout,rec)
            Write (rec,99989) nbaskt
            Call x04baf(nout,rec)
            Write (rec,99988)
            Call x04baf(nout,rec)

            Do i = 1, n
              Write (rec,99987) i, (xbaskt(i,j),j=1,nbaskt)
              Call x04baf(nout,rec)
            End Do

            Write (rec,'()')
            Call x04baf(nout,rec)
            Write (rec,99986)
            Call x04baf(nout,rec)
            Write (rec,'()')
            Call x04baf(nout,rec)
          End If

        End If

        Return

99999   Format (1X,'*** Begin monitoring information ***')
99998   Format (1X,'<Begin displaying search boxes>')
99997   Format (1X,'<End displaying search boxes>')
99996   Format (1X,'Total sub-boxes =',I5)
99995   Format (1X,'Total function evaluations =',I5)
99994   Format (1X,'Total function evaluations used in local search =',I5)
99993   Format (1X,'Total points used in local search =',I5)
99992   Format (1X,'Total sweeps through levels =',I5)
99991   Format (1X,'Total splits by init. list =',I5)
99990   Format (1X,'Lowest level with nonsplit boxes =',I5)
99989   Format (1X,'Number of candidate minima in the "shopping basket','" =', &
          I5)
99988   Format (1X,'Shopping basket:')
99987   Format (1X,'XBASKT(',I3,',:) =',(6F9.5))
99986   Format (1X,'*** End monitoring information ***')
      End Subroutine monit
    End Module e05jcfe_mod
    Program e05jcfe

!     E05JCF Example Main Program

!     This program demonstrates the use of routines to set and get
!     values of optional parameters associated with E05JBF

!     .. Use Statements ..
      Use nag_library, Only: e05jaf, e05jbf, e05jcf, e05jdf, e05jef, e05jff,   &
                             e05jgf, e05jhf, e05jjf, e05jkf, e05jlf, nag_wp,   &
                             x04acf, x04baf
      Use e05jcfe_mod, Only: lcomm, monit, nin, ninopt, nout, objfun, plot
!     .. Implicit None Statement ..
      Implicit None
```

```
!     .. Parameters ..
      Character (*), Parameter             :: fname = 'e05jcfe.opt'
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: loctol, obj
      Integer                              :: i, ibdchk, ibound, ifail, iinit, &
                                              j, loclim, mode, n, n_r, sdlist, &
                                              stclim
      Character (3)                        :: lcsrch
      Character (80)                       :: rec
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: bl(:), bu(:), comm(:),           &
                                              list(:,:), x(:)
      Real (Kind=nag_wp)                   :: ruser(1)
      Integer, Allocatable                 :: initpt(:), numpts(:)
      Integer                              :: iuser(1)
!     .. Intrinsic Procedures ..
      Intrinsic                            :: count, sqrt, trim
!     .. Executable Statements ..
      Write (rec,99992) 'E05JCF Example Program Results'
      Call x04baf(nout,rec)

!     Skip heading in data file
      Read (nin,*)

      Read (nin,*) n, sdlist
      Allocate (bl(n),bu(n),list(n,sdlist),numpts(n),initpt(n),x(n), &
        comm(lcomm))

      Read (nin,*) ibound

      If (ibound==0) Then

!       Read in the whole of each bound

        Read (nin,*)(bl(i),i=1,n)
        Read (nin,*)(bu(i),i=1,n)
      Else If (ibound==3) Then

!       Bounds are uniform: read in only the first entry of each

        Read (nin,*) bl(1)
        Read (nin,*) bu(1)
      End If

      Read (nin,*) iinit

      If (iinit==3) Then

!       User is specifying the initialization list

        Read (nin,*)(numpts(i),i=1,n)
        Read (nin,*)((list(i,j),j=1,numpts(i)),i=1,n)
        Read (nin,*)(initpt(i),i=1,n)
      End If

!     PLOT determines whether MONIT displays information on
!     the current search box:

      Read (nin,*) plot

!     The first argument to E05JAF is a legacy argument and has no
!     significance.

      ifail = 0
      Call e05jaf(0,comm,lcomm,ifail)

!     Open the options file for reading

      mode = 0

      ifail = 0
      Call x04acf(ninopt,fname,mode,ifail)

!     Use E05JCF to read some options from the options file

      ifail = 0
      Call e05jcf(ninopt,comm,lcomm,ifail)

      Write (rec,’()’)
      Call x04baf(nout,rec)

!     Use E05JKF to find the value of the integer-valued option
!     ’Local Searches Limit’

      ifail = 0
      Call e05jkf(’Local Searches Limit’,loclim,comm,lcomm,ifail)

      Write (rec,99999) loclim
      Call x04baf(nout,rec)

!     Compute the number of free variables, then use E05JFF to set the value of
!     the integer-valued option ’Static Limit’

      n_r = count(bl(1:n)/=bu(1:n))
      stclim = 4*n_r

      ifail = 0
      Call e05jff(’Static Limit’,stclim,comm,lcomm,ifail)

!     Use E05JHF to determine whether the real-valued option
!     ’Infinite Bound Size’ has been set by us (in which case
!     E05JHF returns 1) or whether it holds its default value
!     (E05JHF returns 0)

      ifail = 0
      ibdchk = e05jhf(’Infinite Bound Size’,comm,lcomm,ifail)

      If (ibdchk==1) Then
        Write (rec,99998)
        Call x04baf(nout,rec)
      Else If (ibdchk==0) Then
        Write (rec,99997)
        Call x04baf(nout,rec)
      End If

!     Use E05JLF/E05JGF to set the real-valued option
!     ’Local Searches Tolerance’ to the square root of its default

      ifail = 0
      Call e05jlf(’Local Searches Tolerance’,loctol,comm,lcomm,ifail)

      ifail = 0
      Call e05jgf(’Local Searches Tolerance’,sqrt(loctol),comm,lcomm,ifail)

!     Use E05JLF to get the new value of ’Local Searches Tolerance’

      ifail = 0
      Call e05jlf(’Local Searches Tolerance’,loctol,comm,lcomm,ifail)

      Write (rec,99996) loctol
      Call x04baf(nout,rec)

!     Use E05JDF to set the option ’Minimize’ (which is the default)

      ifail = 0
      Call e05jdf(’Minimize’,comm,lcomm,ifail)

!     Use E05JEF to set the option ’Local Searches’ to ’On’ (also
!     the default)

      lcsrch = ’On’
      ifail = 0
      Call e05jef('Local Searches',lcsrch,comm,lcomm,ifail)

!     Get that value of 'Local Searches' using E05JJF

      ifail = 0
      Call e05jjf('Local Searches',lcsrch,comm,lcomm,ifail)

      Write (rec,99995) trim(lcsrch)
      Call x04baf(nout,rec)

!     Solve the problem.

      ifail = 0
      Call e05jbf(n,objfun,ibound,iinit,bl,bu,sdlist,list,numpts,initpt,monit, &
        x,obj,comm,lcomm,iuser,ruser,ifail)

      Write (rec,'()')
      Call x04baf(nout,rec)
      Write (rec,99994) obj
      Call x04baf(nout,rec)
      Write (rec,99993)(x(i),i=1,n)
      Call x04baf(nout,rec)

99999 Format (1X,'Option "Local Searches Limit" has the value ',I6,'.')
99998 Format (1X,'Option "Infinite Bound Size" has been set by us.')
99997 Format (1X,'Option "Infinite Bound Size" holds its default value.')
99996 Format (1X,'Option "Local Searches Tolerance" has the value ',E13.5,'.')
99995 Format (1X,'Option "Local Searches" has the value "',A,'".')
99994 Format (1X,'Final objective value =',F11.5)
99993 Format (1X,'Global optimum X =',2F9.5)
99992 Format (1X,A)
    End Program e05jcfe
```

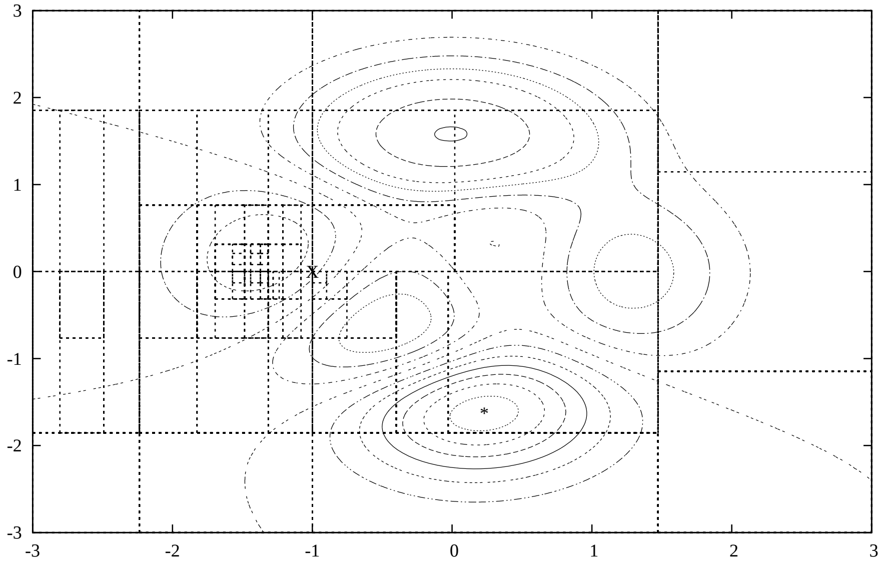
9.2 Program Data

```
E05JCF Example Program Data
  2 3                                                   : N, SDLIST
  0                                                     : IBOUND
  -3.0   -3.0                                           : Lower bounds BL
  3.0   3.0                                             : Upper bounds BU
  3                                                     : IINIT
  3   3   3                                             : NUMPTS
  -3.0   -1.0   3.0   -3.0   0.0   3.0                  : Matrix LIST
  2   2   2                                             : INITPT
  .FALSE.                                               : PLOT


Begin example options file
* Comment lines like this begin with an asterisk
* Set the maximum number of function evaluations
Function Evaluations Limit = 100000
* Set the local search iteration limit
Local Searches Limit = 40
End
```

9.3 Program Results

```
 E05JCF Example Program Results

 Option "Local Searches Limit" has the value     40.
 Option "Infinite Bound Size" holds its default value.
 Option "Local Searches Tolerance" has the value   0.14901E-07.
 Option "Local Searches" has the value "ON".

 (OBJFUN was just called for the first time)

 *** Begin monitoring information ***

 Total sub-boxes =  180
```

```
Total function evaluations =  186
Total function evaluations used in local search =  103
Total points used in local search =    9
Total sweeps through levels =    9
Total splits by init. list =    5
Lowest level with nonsplit boxes =    6
Number of candidate minima in the "shopping basket" =    2
Shopping basket:
XBASKT(  1,:) =  0.22828 -1.34740
XBASKT(  2,:) = -1.62553  0.20452

*** End monitoring information ***


Final objective value =   -6.55113
Global optimum X =  0.22828 -1.62553
```



Example Program
The Peaks Function ***F*** and Search Boxes
The global minimum is denoted by *, while our start point is labelled with X